3/25/2021
Ken Amamori
CS 4375

Producer-Consumer Intro Assignment Report

Instruction:

Work in pairs if possible.

Run pump.py and observe what happens.
Why does the last line never print out?
Change the internal protocol to fix this.

Time it with different buffer sizes.
Which is fastest?
Why?

Think of a way to slightly simplify the code.
For example, you may delete some of the acquire/release lines, or you may move one of those lines, or you may replace some of the "while(count..." loops with simple conditionals.
Now run it a few times and report on whether and how the behavior differ from that of the original code, in terms of both output and time spent.
If the new behavior is different, explain why.
If the new behavior is not different, explain why your new code is equivalent to the old code, or why the test cases you ran were inadequate to demonstrate the difference.

Explanation:

Output before changing.

```
Kens-MBP:producer_consumer_introassignment ken$ python3 pump.py 10
starting Consumer
starting Producer
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classro
```

With the original code of pump.py before changing it, it it will get into infinity loop within the putChar function. This causes because the variable count and bufsize becomes the same value and it satisfies the condition in while-loop. This happens since the producer incremented the count and reaches the size of the buffer.

```
Kens-MBP:producer_consumer_introassignment ken$ python3 pump.py 10
starting Consumer
starting Producer
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.
```

In the 2nd output, which is a modified version of pump.py. Modificaition was made by adding condition variable for charRemoved and charAdded. By using those two variable, two new methods can be introduced, wait() and notify(). In C code, signal will take care of waking up other thread, with this code, notify() will handle that part. With wait() function inside the loop, there won't be needed to have bufLock.release() and bufLock.acquire().

The time of running the python code with differente didn't change or only a few seconds (difference of 0.001s) when there is sufficient buffer size. Time of 'real' changed but that is because of who is running and not relevant to time of the program. When running with '1' buffer size, the user and sys time have incremented by 0.010s. As the number of buffer size increases, the time decrease. However, after the number reaches 5 or 6, the time becomes static, meaning any number greater than 5 or 6 does not affect the time of running the code. I believe it did not change because even though the buffer size has changed, the size of the text hasn't changed. All the space is not being used, therefore even the buffer size has incremented, it did not change. On the other hand, when the buffer size is 1 to 5, the producer will put data into a location where it has been used once, meaning not sufficient place. That is why the time incremented.

Code:

Pump.py (unchanged):

```
1.  #!/usr/bin/env python3
2.  # pump.py  Nigel Ward, 2021.  Based on John Osterhout's Producer/Consumer code.
3.
4.  import sys, threading, time
5.
6.  global count, putIndex,  getIndex, cbuffer, bufLock
7.
8.  def pumpProducer():
9.      print('starting Producer')
10.     arpaciQuote = '"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was
    right but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly here to
    help with that part of your education; after all, when you go to interview at Google, and they ask you a trick
    question about how to use semaphores, it might be good to actually know what a semaphore is, right? But
    Yeats’s larger point is obviously on the mark: the real point of education is to get you interested in
```

```
    something, to learn something more about the subject matter on your own and not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi's dad, Vedat Arpaci) used to say, "Learn beyond the classroom".'
    for charToSend in arpaciQuote:
        putChar(charToSend)


def putChar(character):
    global count, putIndex, bufLock
    bufLock.acquire()
    while count >= bufsize:
        print("waiting to send", end="")
        print(count)
        print(bufsize)
        bufLock.release()
        bufLock.acquire()
    count += 1
    cbuffer[putIndex] = character
    putIndex += 1
    if putIndex == bufsize:
        putIndex = 0
    bufLock.release()

def pumpConsumer():
    print('starting Consumer')
    while (1):
        for i in range(100):
            print(getChar(),end="")
        print("")  # newline

def getChar():
    global count, getIndex, bufLock
    bufLock.acquire()
    while (count == 0):
        print("waiting to receive", end="")
        print(count)
```

```
44.         bufLock.release()
45.         bufLock.acquire()
46.     count -= 1
47.     c = cbuffer[getIndex]
48.     getIndex += 1
49.     if (getIndex == bufsize):
50.         getIndex = 0
51.     bufLock.release()
52.     return c
53.
54. ### main ###
55.
56. if len(sys.argv) != 2:
57.     print(len(sys.argv))
58.     print("usage: pump bufferSize")
59.     exit(1)
60.
61. bufsize = int(sys.argv[1])
62. cbuffer = ['x'] * bufsize    # circular buffer; x means uninitialized
63. count = putIndex = getIndex = 0
64. bufLock = threading.Lock()
65.
66. consumer = threading.Thread(target=pumpConsumer)
67. consumer.start()
68.
69. producer = threading.Thread(target=pumpProducer)
70. producer.start()
71.
72. producer.join()
```

pump.py (modified):

```
1.  #!/usr/bin/env python3
2.  # pump.py  Nigel Ward, 2021.  Based on John Osterhout's Producer/Consumer code.
3.
4.  import sys, threading, time
5.  from threading import Condition
6.
```

```
7.  global count, putIndex,  getIndex, cbuffer, bufLock
8.
9.  def pumpProducer():
10.     print('starting Producer')
11.     arpaciQuote = '"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was right but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly here to help with that part of your education; after all, when you go to interview at Google, and they ask you a trick question about how to use semaphores, it might be good to actually know what a semaphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is to get you interested in something, to learn something more about the subject matter on your own and not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.'
12.     for charToSend in arpaciQuote:
13.         putChar(charToSend)
14.
15.
16. def putChar(character):
17.     global count, putIndex, bufLock, charRemoved, charAdded
18.     bufLock.acquire()
19.     while count >= bufsize:
20.         charRemoved.wait()
21.     count += 1
22.     cbuffer[putIndex] = character
23.     putIndex += 1
24.     if putIndex == bufsize:
25.         putIndex = 0
26.     charAdded.notify()
27.     bufLock.release()
28.
29. def pumpConsumer():
30.     print('starting Consumer')
31.     while (1):
32.         for i in range(100):
33.             print(getChar(),end="")
34.         print("")  # newline
35.
```

```
36. def getChar():
37.     global count, getIndex, bufLock, charRemoved, charAdded
38.     bufLock.acquire()
39.     while (count == 0):
40.         charAdded.wait()
41.     count -= 1
42.     c = cbuffer[getIndex]
43.     getIndex += 1
44.     if (getIndex == bufsize):
45.         getIndex = 0
46.     charRemoved.notify()
47.     bufLock.release()
48.     return c
49.
50. ### main ###
51. if len(sys.argv) != 2:
52.     print(len(sys.argv))
53.     print("usage: pump bufferSize")
54.     exit(1)
55.
56. bufsize = int(sys.argv[1])
57. cbuffer = ['x'] * bufsize    # circular buffer; x means uninitialized
58. count = putIndex = getIndex = 0
59. bufLock = threading.Lock()
60. charAdded = Condition(bufLock)
61. charRemoved = Condition(bufLock)
62.
63. consumer = threading.Thread(target=pumpConsumer)
64. consumer.start()
65.
66. producer = threading.Thread(target=pumpProducer)
67. producer.start()
68.
69. producer.join()
70.
```

Output:

```
Kens-MBP:producer_consumer_introassignment ken$ python3 pump.py 10
starting Consumer
starting Producer
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classro
```

pump.py without changing anything.

```
Kens-MBP:producer_consumer_introassignment ken$ python3 pump.py 10
starting Consumer
starting Producer
"Yeats famously said “Education is not the filling of a pail but the lighting of a fire.” He was rig
ht but wrong at the same time. You do have to “fill the pail” a bit, and these notes are certainly h
ere to help with that part of your education; after all, when you go to interview at Google, and the
y ask you a trick question about how to use semaphores, it might be good to actually know what a sem
aphore is, right? But Yeats’s larger point is obviously on the mark: the real point of education is
to get you interested in something, to learn something more about the subject matter on your own and
 not just what you have to digest to get a good grade in some class. As one of our fathers (Remzi’s
dad, Vedat Arpaci) used to say, “Learn beyond the classroom”.
```

pump.py after modifying.

```
real    0m1.407s
user    0m0.041s
sys     0m0.022s
```

time python3 pump.py 1

```
real    0m1.395s
user    0m0.035s
sys     0m0.016s
```

time python3 pump.py 2

```
real    0m1.376s
user    0m0.034s
sys     0m0.014s
```

time python3 pump.py 4

```
real    0m1.126s
user    0m0.032s
sys     0m0.013s
```

time python3 pump.py 6

```
real    0m1.207s
user    0m0.032s
sys     0m0.013s
```

time python3 pump.py 10

```
real    0m1.003s
user    0m0.031s
sys     0m0.012s
```

time python3 pump.py 100